APPENDIX I

UNIX

TABLE OF CONTENTS

Page

Blank Page

APPENDIX I

UNIX

I.1 Overview. UNIX is a multi-user, multi-tasking operating system. This means more than one person can do more than one operation at the same time.

a. The operating system is the heart of a computer. It accepts input, processes it, and produces output. It handles the interface between the user and the computer. The user enters commands through the terminal, the operating system interprets the commands and distributes the resources needed to complete the job. When data is saved, the operating system finds the space for it.

b. There were two main reasons for creating the UNIX operating system.

(1) First, to create a simple and easy-to-use operating system.

(2) Second, to create a system able to run on diverse platforms.

c. The UNIX operating system has three parts:

(1) *The Kernel*. This is the core of the operating system. It handles security, input and output control, job priorities, and memory management.

(2) *The Shell*. This is a command interpreter. The shell performs jobs in the foreground and background. When issuing commands, the shell interprets the command, determines if the file exists, and lets the kernel know what resources are required to perform the job.

(3) *The Utilities*. This contains text editors, communication capabilities, application programs, and system commands. UNIX stores programs, written text, input data, and UNIX commands in files. UNIX scatters these files throughout its memory. The directory system keeps track of these files so they can be accessed when necessary.

d. The UNIX directory contains a list of subdirectories and files.

(1) The first directory is the home directory. When you are given a user ID, you also get a home directory. This is the starting point of your UNIX session.

(2) Most files are stored in this directory. You may create other directories. UNIX stores additional directories under the home directory.

(3) The UNIX file structure begins with a root directory ('/'), under which the system orders other directories.

(a) Each subdirectory is responsible for a portion of the UNIX operating system.

1 For example, the dev directory contains the addresses of different input and output devices.

2 The bin directory, among other things, contains most of the commands used in UNIX.

3 The etc directory contains the password file.

(b) Each user has a subdirectory within a work subdirectory, such as 'usr'.

(c) There are many other directories, each of which has its own purpose (see figure I.1-1). As you use UNIX, you may create as many directories as you need to organize your data.

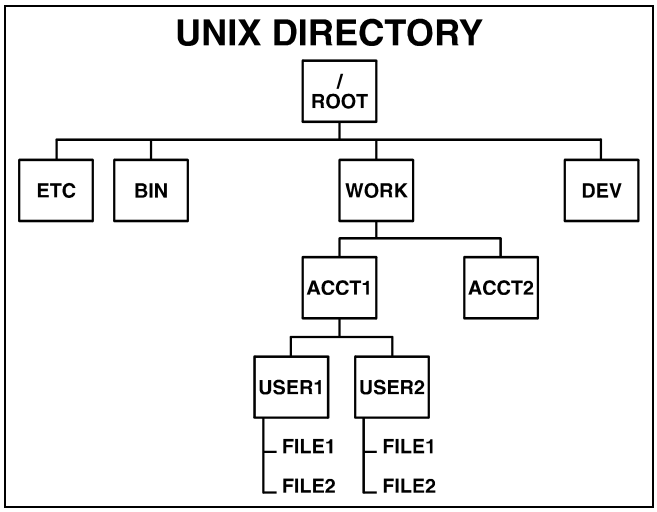

Figure I.1-1. UNIX Directory

e. Knowing how to move from one directory to another is necessary in order to use the system efficiently. To move to another directory, tell the system which directories contain the files you require. The system then follows the pathname you provided. There are two types of pathnames: absolute and relative.

(1) The absolute pathname includes every directory the system goes through to reach a file. All absolute pathnames begin with / (root), and each level is separated by a /. To get to the acct directory, type the full pathname /work/acct. This shows a path from the root directory, through the work directory, to the acct directory. Remember, the / at the beginning of the pathname represents the root directory. The / between the subdirectories represents a different directory level. The absolute pathname is also called the full pathname.

(2) The relative pathname is a shorter pathname relative to the directory you are working in. When you first sign on to the system, you are in your home directory. Assume that is user1. It is also known as the work directory. The current work directory is the directory you are in when you execute a command.

(a) To see your current work directory, type pwd (print working directory) and press <Enter>. The system shows the full pathname of the directory.

(b) If you are looking for a file under the user2 subdirectory, you can change directories and type a full pathname (/work/acct/user2) or a relative pathname.

(c) If you want to go from user1 to user2, change to the directory level that contains both subdirectories. In our example, that is the acct subdirectory. Follow the path to acct and then back to user2.

(d) The two dot symbol (..) means go up one level. Ensure there is a space separating the two dots from the cd command.

(e) Remember to include the / separator between each directory level.

(f) The solution is cd ../user2. This command goes up one level to the acct subdirectory, then down one level to the user2 subdirectory.

f. The following are naming conventions for the UNIX operating system:

(1) Files should have descriptive names to help identify the contents when searching for them.

(2) File names may be up to 14 characters. Acceptable characters include:

(a) Uppercase letters (A-Z).

(b) Lowercase letters (a-z).

(c) Numerals (0-9).

(d) Period (.), comma (,), and underscore(_).

(3) File names may begin with letters or numbers.

(4) File names cannot contain blanks.

(5) File names must be unique to each directory. The same file name may be used for files in different directories.

(6) UNIX is case-sensitive: it distinguishes between a and A.

I.2 UNIX Commands. English syntax is subject-verb-object. UNIX syntax is verb-modifier-object: command-options-object.

a. The command can be any one of hundreds available to UNIX users. A quick reference list of the commands covered in this appendix can be found in table I.2-1. Detailed use of these commands is discussed later in this appendix.

Table I.2-1
UNIX Command Quick Reference

Type	Command	Purpose
Directory	cd	Change directory
	ls	List the contents of a directory
	pwd	Print working directory
File	cat	Concatenate files (type in DOS)
	cp	Copy files
	mv	Move or rename files
	pg	View files with page breaks
	rm	Remove files
	diff	Compare files
Selection	head	Select header lines
	tail	Select trailing lines
	grep	Select lines or rows
	wc	Count characters, words, or lines
System and Processes	who	Tell who is on the system
	ps	Obtain status report on processes
	date	Display system time and date
	man	Obtain the manual pages for a command (Help)
Printing	wpr	Print a file

b. UNIX commands fall into one or more of the following categories: file and directory, selection, combination and ordering, system and processes, printing, security, or bulletin.

c. These commands operate on files, directories, processes, and various devices. It is important to choose the right command for the task at hand.

d. Only the most common commands will be discussed in this appendix. Further information on each command may be obtained by executing the man command, which is explained later in this appendix. Detailed descriptions and uses of the commands are listed in the tables that follow:

(1) Tables I.2-2a through c provide information for each directory command.

(2) Tables I.2-3a through f provide information for each file command.

(3) Tables I.2-4a through d provide information for each selection command.

(4) Tables I.2-5a through d provide information for each system and processes command.

(5) Table I.2-6 provides information for the printing command.

Table I.2-2a
Directory Command: cd

Purpose:	Use the cd command to change directories.
Syntax:	cd [directory]
Usage:	cd lets you change to any other directory that you have permission to work in. Permission is discussed later in this appendix. If only the cd command is keyed, UNIX returns you to the home (or log-in) directory.
Options:	There are no command line options for cd.

Examples:

cd /osc/9130

(This command takes you to the /osc/9130 directory.)

cd osc

(This command takes you to the osc subdirectory of the present working directory. For example, if you were in the /osc/9130 directory and executed the command cd osc, you would now be in the /osc/9130/osc directory.)

Table I.2-2b
Directory Command: ls

Purpose:	Use the ls command to list the contents of a directory.
Syntax:	ls -options file(s) directory(s)
Usage:	ls displays a list of all file names in the working directory. If you use the ls command with a file name, information about that specific file appears. If you use the ls command with a directory name, a list of files and information relative to that directory appears.

Table I.2-2b (Cont.)
Directory Command: ls

Options:

-l Displays a long format list of files.

-C Displays the list of files in multi-column format.

-t Sorts the list by time and date last modified.

-r Reverses the sorting order.

-d Lists only directory names.

Examples:

ls -l /usr/mine

(This command lists, in long format, the contents of the /usr/mine directory. Results may appear as shown below.)

```
-rwxr--r--   1   user  osc   12008    May  27 08:55    makefiles
-rwxrwxrwx 1   user  osc   158955   May  29 13:48    my_docs.001
-rwxrwxr--   1   user  osc   809996   May  31 20:14    my_docs.002
-rwxrwxr--   1   user  osc   22773    Apr  01 11:26    notes
drwxrwxr--  2   user  osc   96       May  27 07:44    usage.ace
-rwxrwxr--   1   user  osc   22997    Apr  02 07:19    usage.arc
-rwxrw----   1   user  osc   93446    Jan  09 16:77    usage.rpt
```

NOTE: Files appear in alphabetical order.

Examples (Cont.):

ls -ltr /usr/mine

(This command gives the same results as the ls -l command except that the list appears in reverse date and time order so the newest files are first.)

Table I.2-2c
Directory Command: pwd

Purpose: The pwd command prints the present working directory.

Syntax: pwd

Usage: This is the easiest way to identify where you are in the file system

Options: There are no command line options for pwd.

Example: pwd
/usr/mine

(This command is helpful when you have been working in many different directories.)

Table I.2-3a
File Command: cat

Purpose: The cat command displays contents of files. It may also join files.

Syntax: cat -options files

Usage: The cat command is one way to display data in a file. The command shows the entire file on the screen without pauses. It is similar to the type command in MS-DOS.

Options: Refer to the system man pages for more options.

Example: cat my_notes

(This command displays the entire contents of the file my_notes.)

Table I.2-3b
File Command: cp

Purpose: The cp command copies files.

Syntax: cp -options file newfile

or

cp -options file directory

Usage: Use the cp command to back up files or copy them from one directory to another. When files are copied, file permissions are also copied.

Options: Refer to the system man pages for more options.

Examples:

cp my_notes my_notes.bak

(This command creates a copy of the file my_notes and names it my_notes.bak in the working directory.)

cp /usr/usr1/todays_notes /usr/mine

(This command copies the file todays_notes from the /usr/usr1 directory to the /usr/mine directory.)

Table I.2-3c
File Command: mv

Purpose: Use the mv command to move or rename files.

Syntax: mv -options file newfile

or

mv -options files directory

Usage: The mv command is a useful organization tool. It allows you to move or rename files to better suit your needs.

Options: Refer to the system man pages for more options.

Examples:

mv results.out results.may_27

(This command renames the file results.out as results.may_27 in the working directory.)

mv results.may_27 /test/results

(This command moves the file results.may_27 from the working directory to the /test/results directory.)

NOTE: Unlike the cp command, when you use mv, the file no longer exists in the working directory.

Table I.2-3d
File Command: pg

Purpose: Use the pg command to view a file one screen at a time.

Syntax: pg -options files

Usage: Use the pg command to scan files larger than one screen. The default viewing size is 24 lines per page. Each page ends with a colon (:).

Options: Refer to the system man pages for command line arguments. The following options can be executed from the colon (:) prompt after the pg command is executed:

<Enter>	Advances one screen.
+10	Advances ten screens.
-7	Moves back seven screens.
/pattern	Moves forward to the first occurrence of the pattern.

Options (Cont.):

q	Quits viewing files.
n	Moves to the next file (used when viewing multiple files).
p	Moves back one file.

Example: pg log.001 log.002 log.003

(This command shows the files log.001, log.002, and log.003 in the order they were entered on the command line.)

Table I.2-3e
File Command: rm

Purpose: Use the rm command to remove or delete files.

Syntax: rm -options files.

Usage: rm purges unneeded files.

Options:

-i Prompts you before removing each file selected on the command line.

-r Accepts directories as an argument. If you use the -r option and the directory is empty, that directory is removed.

Examples:

rm -i *.00*

(This removes all files with .00 in the file name. The -i option tells the system to prompt you before removing each file.)

Examples (Cont.):

rm -r /user/user1/back/*

(This removes all files from the /user/user1/back directory and also removes the back directory.)

Table I.2-3f
File Command: diff

Purpose: diff compares two or more files.

Syntax: diff -options files

Usage: diff compares files and displays line-by-line differences.

Option: -b Ignores trailing blanks at the end of the lines.

Example: diff log.001 log.bak

(This compares the files log.001 and log.bak. If differences are found, they are shown. If the files are identical, the system returns to the prompt.)

Table I.2-4a
Selection Command: head

Purpose: head views the top portion of a file.

Syntax: head -options file.

Usage: head lets you view a selected portion of a file. The default is the first 10 lines.

Options:

-number Displays the first [number] of lines from the top of the file.

+number Shows the number of lines selected beginning at the top of the file.

Examples:

head -20 my_notes

(This command shows the first 20 lines of the file my_notes.)

head my_notes

(This command shows the first 10 lines of the file my_notes.)

Table I.2-4b
Selection Command: tail

Purpose: tail views the bottom portion of a file.

Syntax: tail -options file.

Usage: tail also lets you view a selected portion of a file. The default for tail is the last 10 lines.

Options:

-number Shows the number of lines selected above the end of the file.

Examples:

tail my_notes

(This command shows the last 10 lines of the file my_notes.)

Examples (Cont.):

tail -20 my_notes

(This command displays the last 20 lines of the file my_notes.)

Table I.2-4c
Selection Command: grep

Purpose: grep searches files for lines containing a selected pattern.

Syntax: grep -options pattern files.

Usage: grep finds selected items or information in files.

Options:

-v Shows those lines that do not match the given pattern.

-c Shows a count of the lines that match the given pattern.

Examples:

grep "(313)" phone_dir

(This command shows all lines in the file phone_dir that contain the pattern (313).)

grep -v "(817)" phone_dir

(This command shows all lines in the file phone_dir that do not contain the pattern (817).)

Table I.2-4c (Cont.)
Selection Command: grep

Examples (Cont.):

grep -c "(817)" phone_dir

(This command shows a count of lines containing the pattern (817) in the file phone_dir.)

grep "May 22" log.001

(This command shows all lines in the file log.001 containing the pattern May 22.)

NOTE: When there is a blank space in the pattern to be searched for, the pattern must be enclosed in quotes ("May 22").

Table I.2-4d
Selection Command: wc

Purpose: wc counts characters, words, and lines in a file.

Syntax: wc -options files.

Usage: wc determines file sizes by characters, words, and lines. If the wc command is used with no options, the number of characters, words, and lines in a file appears.

Options:

-l Shows the line count.

-w Shows the word count.

-c Shows the character count.

Table I.2-4d (Cont.)
Selection Command: wc

Examples:

wc log.001

(This command shows the number of characters, words, and lines in the file log.001.)

wc -wl trans.227

(This command shows the number of words and lines in the file trans.227.)

Table I.2-5a
System and Processes Command: who

Purpose: who is used to find out who is logged on to the system.

Syntax: who -options

Usage: Use who to see if a particular person or activity is using the system. Also use who to check the general level of activity. The who command shows the following:

NAME	Tells the user's log-in name.
STATE	Tells if the user can be written to.
LINE	Tells the line or terminal in use.
TIME	Tells the time and date the user logged in.
IDLE	Tells the length of time the user has been inactive.
PID	Tells the user's process ID number.

Table I.2-5a (Cont.)
System and Processes Command: who

COMMENT Displays comments when loaded.

EXIT Tells the exit values of dead processes.

Options:

-u Lists those users currently logged in.

-H Displays column headings.

-q Displays a quick list of those logged in.

Examples:

who

(This tells you who is logged on to the system. The output may appear as shown below.)

```
root     console   May 22  07:08
jbgood   tty819    May 22  08:19
smith    ttyp002   May 22  10:47
jones    ttyp003   May 22  13:58
```

who -uh

(This tells you who is currently logged on to the system and also shows complete information and column headings. The output may appear as shown below:)

NAME	LINE	TIME	IDLE	PID
root	console	May 22 07:08	.	17
jbgood	tty819	May 22 08:19	07.11	12
smith	ttyp002	May 22 10:47	0.13	221
jones	ttyp003	May 22 13:58	.	223

NOTE: The (.) in the IDLE column means the line is active.

Table I.2-5b
System and Processes Command: ps

Purpose: ps obtains a status report of processes.

Syntax: ps -options

Usage: ps lets you identify process status.

Options: ps has many options. Two of the most common ones are listed below. Refer to the system man pages for a complete listing of options.

-ef Generates a full list of all processes currently running.

-fu userid Displays a process report for a specific user.

Examples:

ps -ef

(This gives a full list of all active processes.)

ps -fu user7

(This shows all processes being run by the person with the log-in ID user7.)

Table I.2-5c
System and Processes Command: date

Purpose: date obtains the current date and time.

Syntax: date +format

Usage: The date command checks the current date and time.

Options: +%j shows the Julian date.

Table I.2-5c (Cont.)
System and Processes Command: date

Examples:

date

(This shows the system date and time. Output is in the format below.)

Tue May 22 01:27:00 CST 1992

date +%j

(This command shows the Julian date.)

Table I.2-5d
System and Processes Command: man

Purpose: man shows the manual pages that explain the UNIX commands.

Syntax: man command

Usage: Use man when you want more information about a UNIX command or command options.

Options: Refer to the system man pages for a complete listing of options.

Example: man grep

(This shows the manual pages for the grep command. It contains a description of the command, how to use it, and the command line options.)

Table I.2-6
System and Processes Command: wpr

Purpose: wpr prints files. (unique to SARSS-GW)

Syntax: wpr files

Usage: wpr can print one or many files.

Options: Refer to the system man pages for a complete listing of options.

Example: wpr my_notes

(This command prints a copy of the file my_notes on the user's printer.)

NOTE: The command wpr is not compatible with all printers.

e. Many command line arguments can be used in conjunction with the commands listed above. Several of these are:

(1) The pipe ¦ utility.

(a) The ¦ can combine several commands on one command line. For example:

ls -ltr ¦ pg

(b) The commands are interpreted left to right. The first command shows a directory list in the long format, order reverse, by date and time (ls -ltr). Because the pg follows that command, the output shows one screen at a time. The ¦ sign may be used many times on single command line.

pg log.001 ¦ grep "May 31" ¦ grep -v comment

(c) This shows the file log.001 one screen at a time (pg log.001). The ¦ sign means the only lines shown are those containing the pattern May 31 (grep "May 31"). The next ¦ trims the list by limiting it to those previously stated lines that do not contain the word comment (grep -v comment). The entire command

shows only those lines that contain the pattern May 31 and do not contain the pattern comment from the log.001 file.

(2) The * and ? (wild cards).

(a) UNIX interprets the * as any character string.

ls -l log.*

(b) This command shows a list of files in the present working directory that begin with log. The * may be used more than once on the command line. The example below shows a long list of files with the characters rp in the file names.

ls -l *rp*

(c) The ? matches any single character. For example, the following command copies all files in the present working directory that match logs."any single character"02.

cp logs.?02 /temp

(3) The > (redirect).

(a) The > redirects the normal screen output into a file. For example, the command to retrieve all lines in the file log.001 that contain the character string May 31 and put them into a new file is:

grep "31 May" log.001 > newfile

(b) The redirection sign > generates the file newfile with only the information that was requested.

NOTE: When using the >, nothing is sent to the screen.

I.3 UNIX File and Directory Permissions. All files and directories in UNIX have permissions. The permissions of a file or directory not only determine who has the right to access that file or directory, but also to what level they may access it. Permissions also show whether a file can be executed or a directory traversed.

a. There are three types of permissions for three different areas:

r	Lets you read a file or access a directory.
w	Lets you create or modify a file.
x	Lets you execute a file.

b. These permissions are applied to three categories of users:

owner	Whoever owns a file.
group	Whatever group the user belongs to.
global	Whoever is on the system.

c. These two categories combine to form the permission designations seen when the ls -l command is run.

ls -l /usr/mine

```
drwxrwxr-- 2 smith  osc  96                  May  27 07:44  notes
-rwxr--r--   1 jones  osc  12008 May  27 08:55  makefile
-rwxrwxr-- 1 jdoe   osc  98722 Jun  18  13:55  users.prg
```

(1) This lists, in long format, the contents of the /usr/mine directory. File permissions are indicated in the left column.

(2) The first position in the permission column tells whether or not the entry is a directory, file, or link. If the first position is d, the entry is a directory. If the first position is -, the entry is a file.

(3) The other nine positions are the permissions for that file or directory. The third column in the display identifies the file owner. The fourth column identifies the group the owner belongs to. For example, the first item listed after the ls -l command was executed:

```
PERMISSIONS     LINKS    OWNER     GROUP

drwxrwxr--          2       smith       osc  96  May  27 07:44  notes
```

(a) The owner of this directory is smith.

(b) The group is osc.

(c) The file permissions are rwxrwxr--.

1 These permissions are broken into three blocks of three letters:

OWNER	GROUP	GLOBAL
rwx	rwx	r--

2 This means smith has read, write and execute permissions. Also, any user belonging to the osc group has read, write, and execute permissions. The last three characters indicate any user, regardless of who they are or what group they belong to, has read permissions only. The - indicates permission is not available.

I.4 UNIX Editors. Several types of editors are contained within the UNIX operating system. One, a line editor, is called ed. It cannot do full-screen editing. Changes are made from the command line.

a. Ed is inefficient and cumbersome.

b. The vi editor is better. The vi editor allows full-screen editing to create and modify files. All commands and cursor movements use typewriter keys for execution.

(1) To create a file in vi, enter the vi command and a file name. If the file name already exists, vi will retrieve the file and show it on the screen. If the name is a new file, the screen will look like the one in figure I.4-1.

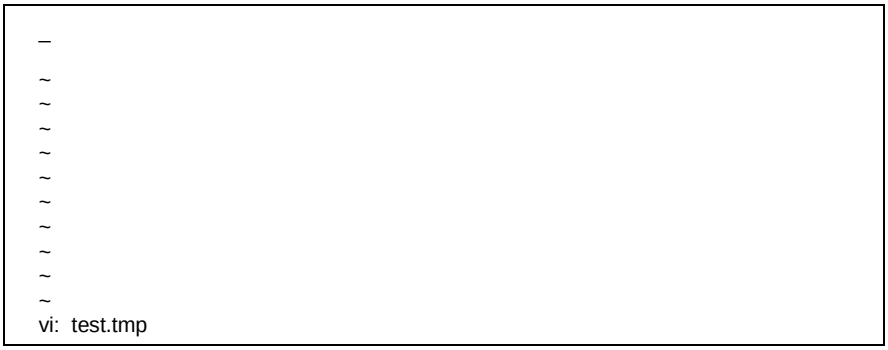

Figure I.4-1. vi Edit Screen

(2) The cursor is on the first line and the tildes represent blank lines. The last line of the screen is the information status line. If any commands are wrong, an error message appears on this line.

(3) When using vi, there is a specific sequence for entering the commands.

(a) Press <Esc>. This tells vi you are entering a command, or that you are finished with the input mode.

(b) Position the cursor where you wish to edit. The following keys move the cursor in the directions shown:

NOTE: Do not use the arrow keys to move the cursor.

(c) Enter the command to be executed. Be careful when typing because vi is case-sensitive. The uppercase J means something different than the lowercase j.

(d) Do appropriate data entry. When finished, press <Esc> to let the system know you are finished.

c. To enter data, turn on one of the input modes: insert, append, change, or replace.

(1) To append, press the <a> key. The <a> key appends the data after the cursor. An A inputs data after the last character on the line.

(2) An i inserts data immediately before the cursor. An I inserts data at the beginning of the line.

(3) An o opens a blank line immediately below the current line. An O opens a blank line immediately above the current line.

(4) Press <Esc> after entering all the data when using these commands.

d. There are many ways to save data in vi.

(1) To save the data as the original file name, enter ZZ. This saves the file and exits to the shell.

(2) To save a file and remain in the editor, enter :w. Use this periodically to save your work.

(3) To save the file under another file name, enter :w <filename>.

(4) If you want to get out without saving the file, enter :q!.

e. Many cursor-movement commands do more than move lines or characters. Listed below in table I.4-1 are some of those commands.

Table I.4-1
Cursor Commands

Command	Action
H	Moves the cursor to the home position (the first position of the first line).
G	Moves the cursor to the last line of the file.
w	Moves the cursor forward one word.
b	Moves the cursor back one word.
(	Moves the cursor to the beginning of the current sentence.
)	Moves the cursor to the beginning of the next sentence.
{	Moves cursor to the beginning of current paragraph.
}	Moves the cursor to the beginning of the next paragraph.
$	Moves the cursor to the end of current line.
0	Moves the cursor to the beginning of the current line.
<ctrl d>	Scrolls down 12 lines.
<ctrl u>	Scrolls up 12 lines.

(1) Two spaces distinguish sentence beginnings and endings for UNIX.

(2) H returns the cursor to the first position of the first line on the screen.

(3) G moves the cursor to the last line of the file.

(4) The w moves the cursor forward one word

(5) The b moves the cursor back one word.

(6) The left parenthesis (moves the cursor to the beginning of a sentence.

(7) The right parenthesis) moves the cursor to the end of the sentence.

(8) The left brace { move the cursor to the beginning of a paragraph.

(9) The right brace } moves the cursor to the end of a paragraph.

f. To find a specific character string in the file, enter the desired string preceded by a /. The cursor moves to the string.

(1) Enter //, if you desire a second occurrence of that string.

(2) Another search method is to enter an n to continue a search in the same direction. An N reverses the search direction.

g. Many of the commands typed on the screen are not shown and it is easy to make a mistake. Because of this possibility, vi has an undo or u command. The u command returns the file to just before the last command was entered. The undo command works only on the last command entered.

h. A couple commands delete or change text.

(1) The d deletes text.

(2) The c changes text.

(3) Using these commands requires a range of changes or a range of deletions. Table I.4-2 shows the range used after the c or d command:

Table I.4-2
UNIX Range Commands

Range	Action
e	From the cursor to end of word.
w	From the cursor to the beginning of the next word, including space.
b	From the letter before the cursor backward to the beginning of the word.
(	From the character before the cursor backwards to the beginning of the current sentence.
)	From the cursor forward to the beginning of the next sentence.
{	From the character before the cursor backward to the beginning of the current paragraph.
}	From the cursor to the end of the paragraph.
$	From the cursor to the end of the current line.
0	From the character before the cursor to the beginning of the current line.

(4) To delete the current line of text, enter dd.

(5) To delete one character only, place the cursor under the character and press <x>.

(6) To transpose letters such as the, place the cursor under the e and press <xp>.

Blank Page